KI-gestützte Entwicklung

VON DER IDEE ZUR PRODUKTION

Der komplette All-in-One-Leitfaden, um deine eigene Website, App oder dein Startup von Grund auf zu erstellen. Lerne KI-Agenten und Tools zu beherrschen, Zahlungen zu integrieren, deinen Server bereitzustellen und Marketing-Tricks für organisches Wachstum.

apifreellm.com

Version 1.0 — Februar 2026

Inhaltsverzeichnis

1. Einführung

> *Du liest das gerade, weil irgendwo eine Kombination aus Marketing-Strategien, SEO-Techniken und cleverer Positionierung diesen Kurs auf deinen Bildschirm gebracht hat. Das allein sollte dir etwas sagen: Die Taktiken in diesem Leitfaden funktionieren tatsächlich. Und ja, du wirst jede einzelne davon lernen.*

Die Welt hat sich verändert

Softwareentwicklung ist nicht mehr das, was sie einmal war. Vor ein paar Jahren erforderte der Aufbau einer Webanwendung monatelange Arbeit, ein Team von Entwicklern und ein beträchtliches Budget. Heute kann eine einzelne Person mit den richtigen Tools und dem richtigen Wissen eine produktionsreife Anwendung in Tagen statt Monaten erstellen und veröffentlichen.

Dieser Kurs wurde von Fachleuten geschrieben, die in der Branche arbeiten und täglich KI-Agenten-Tools verwenden, um echte Produkte zu bauen. Wir unterrichten keine Theorie aus dem Lehrbuch. Wir lehren, was in der realen Welt tatsächlich funktioniert, genau jetzt.

Warum es diesen Kurs gibt

KI und LLMs sind mittlerweile bessere und schnellere Umsetzer als Menschen. Sie können Code schreiben, debuggen, refaktorisieren und deployen mit einer Geschwindigkeit, die kein Mensch erreichen kann. Aber etwas fehlt ihnen grundlegend: **Ideen**.

> KI-Modelle sind außergewöhnliche Umsetzer, aber sie sind keine Innovatoren. Sie sehen keine Marktlücke und denken „Da könnte ich etwas bauen, um das zu lösen." Das ist dein Job. Deine Rolle ist es, der Architekt der Ideen zu sein. Die KI ist dein Baumeister.

Wenn du einem LLM schlechte Anweisungen gibst, wird es trotzdem etwas produzieren. Es wird nicht anhalten und erklären, was tatsächlich besser wäre. Die Qualität dessen, was du baust, ist direkt proportional zur Qualität deines Wissens. Dieser Kurs schließt diese Lücke.

Das ist nicht nur ein Entwicklungskurs. Das ist eine komplette Blaupause, um von einer Idee zu einem live geschalteten, umsatzgenerierenden Produkt zu gelangen. Einschließlich extrem effektiver Marketing- und SEO-Taktiken — dieselben Techniken, die dich auf genau diese Seite gebracht haben.

2. Der KI-First-Entwicklungsstack

> *Die Tools, die du wählst, bestimmen, wie schnell du vorankommst. In diesem Kapitel analysieren wir die heute verfügbaren KI-Coding-Agenten, helfen dir, den richtigen auszuwählen, und lehren dich die Prompting-Strategien, die Amateure von Profis unterscheiden.*

> **Hinweis:** Dieser Leitfaden wurde unter Windows geschrieben, aber alle Tools und Schritte funktionieren identisch auf macOS und Linux. Google Antigravity, Claude Code und jede andere in diesem Kurs besprochene Anwendung sind vollständig plattformübergreifend. Wenn du einen Mac oder eine Linux-Maschine verwendest, folge einfach denselben Schritten — die Oberflächen und Workflows sind identisch.

Die KI-Agenten-Landschaft

Der Bereich der KI-Coding-Tools hat sich explosionsartig entwickelt. Aber nicht alle Tools sind gleich. Einige sind glorifizierte Autovervollständigungs-Engines. Andere sind vollautonome Agenten, die deine gesamte Codebasis lesen, eine Strategie planen, Änderungen über mehrere Dateien hinweg ausführen, Tests durchführen und Fehler selbstständig beheben können. Diesen Unterschied zu verstehen ist entscheidend.

Es gibt zwei grundlegende Kategorien von KI-Coding-Tools:

- **Inline-Assistenten** — Diese sitzen in deinem Editor und schlagen Code vor, während du tippst. Denk an den ursprünglichen GitHub Copilot. Sie sind reaktiv: Sie warten, bis du schreibst, und versuchen dann zu erraten, was als Nächstes kommt. Nützlich, aber begrenzt.
- **Agentische Tools** — Diese sind eine völlig andere Liga. Du gibst ihnen eine Aufgabe in natürlicher Sprache, und sie planen, schreiben, bearbeiten, debuggen und iterieren autonom. Sie schlagen nicht nur eine Codezeile vor. Sie bauen Features. Hier liegt die wahre Power.

Hier sind die Tools, die gerade wichtig sind:

Claude Code (von Anthropic)

Claude Code ist unserer Erfahrung nach der leistungsfähigste KI-Coding-Agent, der heute verfügbar ist. Es ist ein terminalbasierter Agent, der direkt in deinem Projektverzeichnis arbeitet. Du gibst ihm Anweisungen in natürlicher Sprache, und er liest deine Dateien, schreibt Code, führt Befehle aus, erstellt Commits und behebt Fehler autonom. Er hat vollen Zugriff auf dein Dateisystem und deine Shell, was ihn unglaublich effektiv für echte Entwicklungsarbeit macht. Du kannst ihn über npm installieren (`npm install -g @anthropic-ai/claude-code`) oder als VS Code-Erweiterung verwenden, die wir gleich besprechen werden.

Google Antigravity

Antigravity ist eine Entwicklungsumgebung von Google, die KI-gestützten Chat und Agenten-Fähigkeiten direkt in deinen Workflow bringt. Stell es dir als einen intelligenten Arbeitsbereich vor, in dem du über Chat-Interfaces mit KI-Agenten interagieren kannst, und aus jeder Agenten-Konversation kannst du einen integrierten VS Code-Editor öffnen. Das ist der Schlüssel: Antigravity gibt dir die konversationelle KI-Schicht, während VS Code die Code-Editing-Power liefert. Die Kombination ist nahtlos — du besprichst mit dem Agenten, was du bauen willst, und springst dann direkt in den Code, ohne Fenster wechseln zu müssen.

Cursor

Cursor ist ein Fork von VS Code mit tief integrierter KI. Es hat sowohl Inline-Vorschläge als auch einen agentischen „Composer"-Modus, in dem du Änderungen über mehrere Dateien hinweg beschreiben kannst. Die Benutzeroberfläche ist vertraut, wenn du bereits VS Code verwendest, und es unterstützt mehrere Modelle (Claude, GPT usw.). Es ist ein solides Tool, besonders wenn du einen vollständig visuellen Workflow bevorzugst. Allerdings sind seine agentischen Fähigkeiten, obwohl gut, nicht so autonom wie Claude Code. Du musst oft einzelne Änderungen Schritt für Schritt überprüfen und genehmigen.

> Unser empfohlenes Setup: **Google Antigravity + Claude Code als VS Code-Erweiterung.** Nutze Antigravitys Agenten-Chats zum Planen und Besprechen deiner Arbeit, öffne dann den integrierten VS Code und lass Claude Code die schwere Umsetzung übernehmen. Das gibt dir das Beste aus beiden Welten: intelligente Konversation für die Planung und autonome Code-Ausführung für den Aufbau.

Deinen Agenten wählen & konfigurieren

Ein agentisches Tool zu installieren und auszuführen ist der einfache Teil. Das Maximum herauszuholen erfordert ein Verständnis dafür, wie es funktioniert, das richtige Abonnement zu wählen und es korrekt einzurichten.

Das Claude-Abonnement: starte mit Pro

Claude Code erfordert einen Anthropic-Account. Wir empfehlen, mit dem **Claude Pro-Abonnement** zu starten, der Einstiegs-Bezahlstufe. Es ist erschwinglich und gibt dir Zugang zu Claude Code mit allen Features. Es ist der perfekte Ausgangspunkt zum Experimentieren, den Workflow zu lernen und dein erstes einfaches Projekt zu bauen.

Beachte jedoch, dass der Pro-Plan ein **begrenztes Nutzungskontingent** hat. Wenn du Claude Code intensiv nutzt, erreichst du das Limit relativ schnell. Zum Lernen und für kleine Projekte reicht es mehr als aus. Wenn du aber bereits weißt, dass du Claude Code als dein primäres Entwicklungstool verwenden und täglich stundenlang damit arbeiten willst, ziehe ein Upgrade auf einen höheren Plan (wie Max) von Anfang an in Betracht. Die höheren Pläne bieten deutlich mehr Nutzungskontingent, was weniger Unterbrechungen und einen flüssigeren Workflow beim Erstellen größerer Projekte bedeutet.

Das Modell: Claude Opus 4.6

Wir empfehlen derzeit **Claude Opus 4.6** als dein primäres Entwicklungsmodell. Es ist aktuell das beste Modell für den Aufbau von Websites und Anwendungen. Es versteht komplexe Architekturen, schreibt sauberen und produktionsreifen Code, bewältigt Änderungen über mehrere Dateien mit bemerkenswerter Genauigkeit und braucht selten Korrekturen beim ersten Versuch. Wenn du mit Claude Code arbeitest, stelle Opus 4.6 als dein Standardmodell ein. Der Unterschied in der Ausgabequalität im Vergleich zu kleineren Modellen ist sofort spürbar.

Antigravity einrichten

Ab diesem Punkt fährt dieser Leitfaden mit **Google Antigravity** als unserer primären Entwicklungsumgebung fort. So machst du alles startklar.

Schritt 1: Erstelle deinen Projektordner. Bevor du Antigravity öffnest, erstelle einen Ordner auf deinem Computer, in dem dein erstes Projekt leben wird. Zum Beispiel könntest du unter Windows `C:\Projects\my-first-app` erstellen, oder auf Mac/Linux `~/Projects/my-first-app`. Das ist der Ordner, den Antigravity als Workspace öffnen wird. Er kann vorerst leer sein — wir füllen ihn später im Kurs mit Code.

Schritt 2: Installiere und starte Antigravity. Lade Google Antigravity herunter, installiere es und öffne es. Melde dich mit deinem Google-Konto an, wenn du dazu aufgefordert wirst.

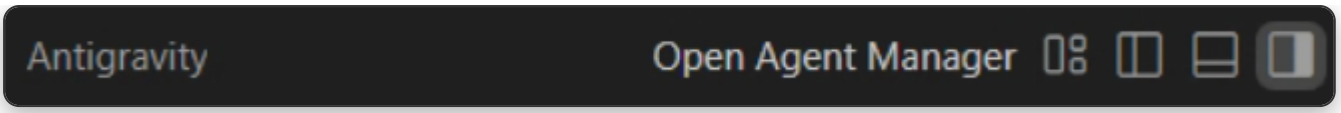
Während des Installationsprozesses wird Antigravity dich bitten, einige Richtlinien zu konfigurieren. Für einen schnellen, autonomen Entwicklungs-Workflow empfehlen wir, die **benutzerdefinierte Konfiguration** auszuwählen und diese Optionen einzustellen:

- **Terminal-Ausführungsrichtlinie** → Immer fortfahren
- **Überprüfungsrichtlinie** → Immer fortfahren
- **JavaScript-Ausführung** → Immer fortfahren

Mit diesen Einstellungen können die Agenten von Antigravity Befehle ausführen, Dateien schreiben und Code autonom ausführen, ohne bei jedem Schritt nach Bestätigung zu fragen. Das ist es, was die schnelle, flüssige Entwicklungserfahrung ermöglicht, die wir in diesem Kurs anstreben.

> **Sicherheitswarnung:** Wenn du Agenten erlaubst, autonom fortzufahren, können sowohl Antigravity als auch Claude Code Aktionen auf deinem System ohne manuelle Genehmigung ausführen. Das bedeutet, du musst darauf achten, was du ihnen zugänglich machst. Richte Agenten nicht auf Codebasen, die Malware enthalten könnten, und sei vorsichtig mit Links oder Ressourcen aus nicht vertrauenswürdigen Quellen. Im KI-Zeitalter gibt es neue Angriffsvektoren — böswillige Akteure können Inhalte erstellen, die speziell darauf ausgelegt sind, LLMs dazu zu bringen, schädliche Befehle auszuführen. Behalte immer im Auge, was deine Agenten tun. Wir empfehlen das „Immer fortfahren"-Setup für Geschwindigkeit, aber bleib wachsam und überprüfe die Ausgabe regelmäßig.

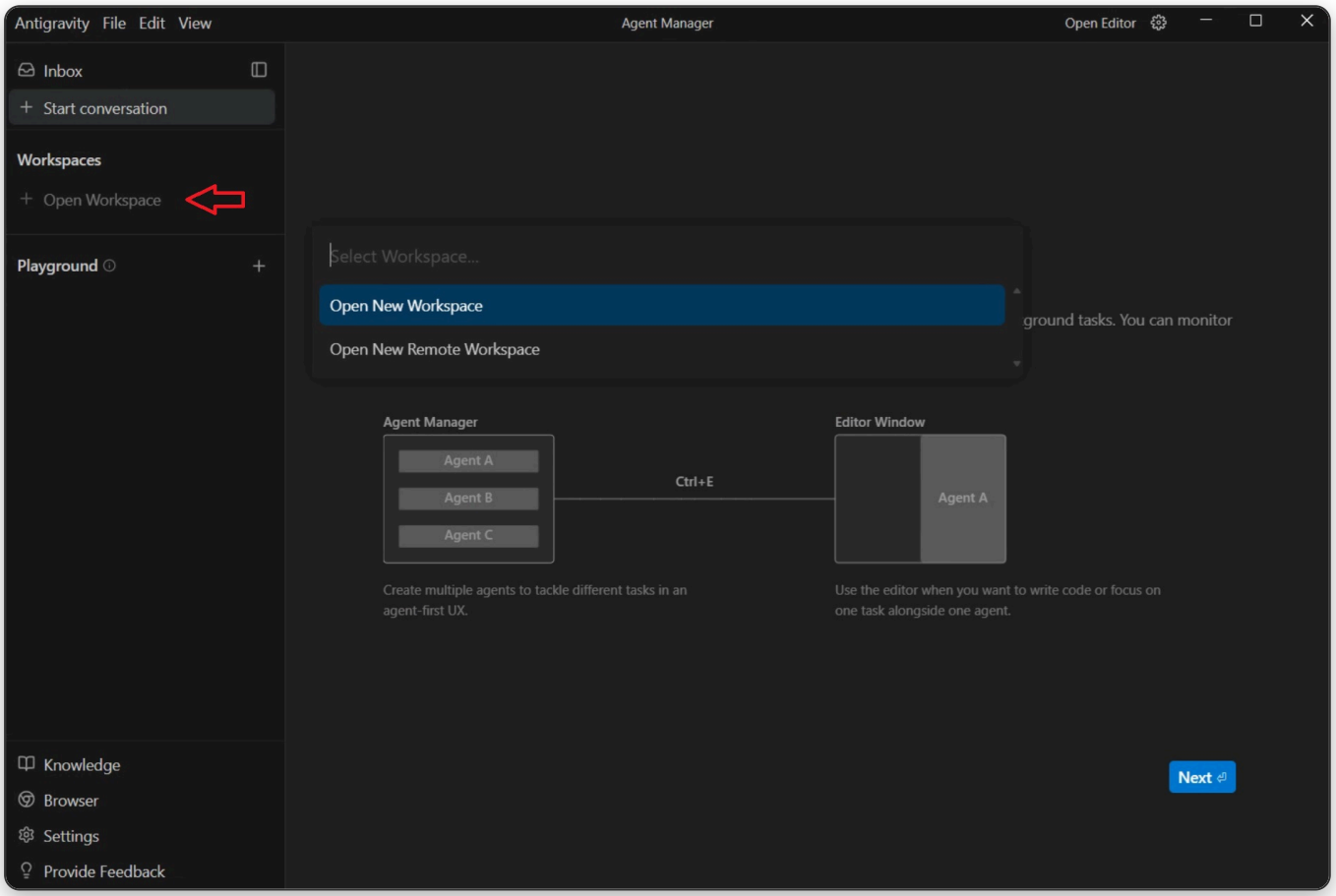
Schritt 3: Öffne den Agent Manager. Sobald Antigravity läuft, klicke auf **„Open Agent Manager"** in der oberen Leiste des Fensters.



Der „Open Agent Manager"-Button in Antigravitys oberer Leiste.

Der **Agent Manager** ist die zentrale Anlaufstelle von Antigravity. Hier verwaltest du deine Projekte, startest KI-Konversationen und öffnest deine Entwicklungs-Workspaces.

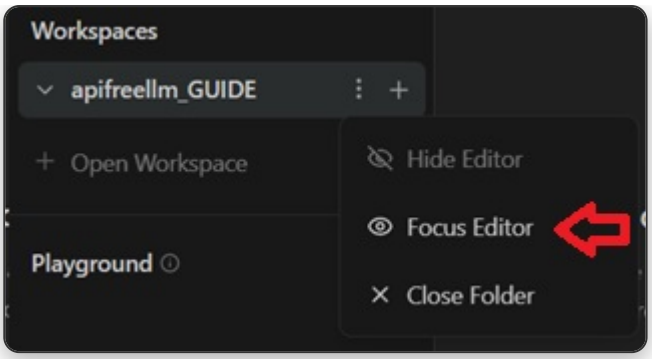
Schritt 4: Erstelle deinen ersten Workspace. Im Agent Manager schau dir die linke Seitenleiste an. Unter dem Abschnitt **„Workspaces"** klicke auf **„+ Open Workspace"**. Ein Dropdown erscheint — wähle **„Open New Workspace"**.



Klicke auf „+ Open Workspace" in der linken Seitenleiste, dann wähle „Open New Workspace".

Antigravity wird dich bitten, einen Ordner auszuwählen. Navigiere zum Projektordner, den du in Schritt 1 erstellt hast, und wähle ihn aus. Dein neuer Workspace erscheint in der linken Seitenleiste unter „Workspaces" mit dem Namen des ausgewählten Ordners. Stell dir jeden Workspace als eine individuelle Entwicklungssitzung vor — einen pro Projekt. Du kannst so viele Workspaces erstellen, wie du brauchst, und jederzeit im Agent Manager zwischen ihnen wechseln.

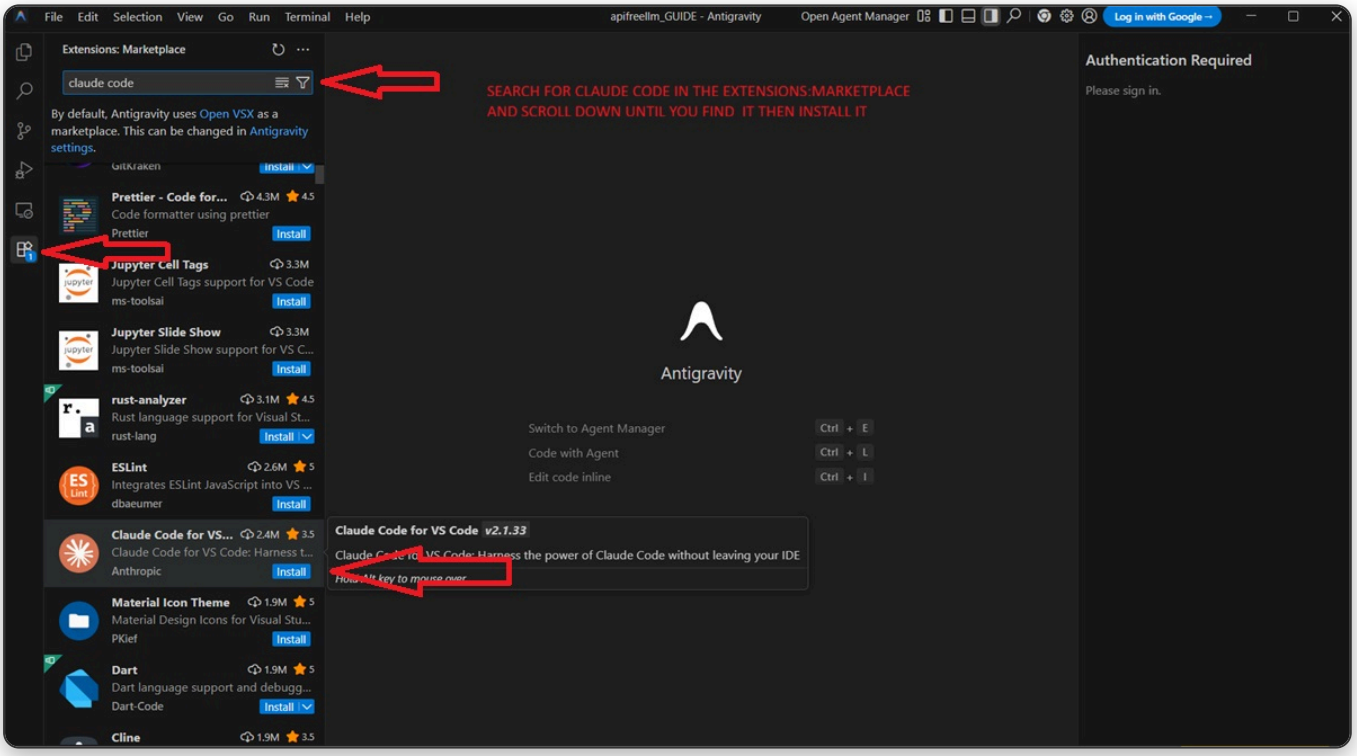
Schritt 5: Öffne den Editor. Sobald dein Workspace erstellt ist, siehst du seinen Namen in der Seitenleiste. Klicke auf die **drei vertikalen Punkte** (:) neben dem Workspace-Namen und wähle dann **„Focus Editor"**. Das öffnet die vollständige VS Code-Umgebung für diesen Workspace, in der du deinen Code schreibst und bearbeitest.



Klicke auf die drei Punkte neben deinem Workspace-Namen und wähle „Focus Editor".

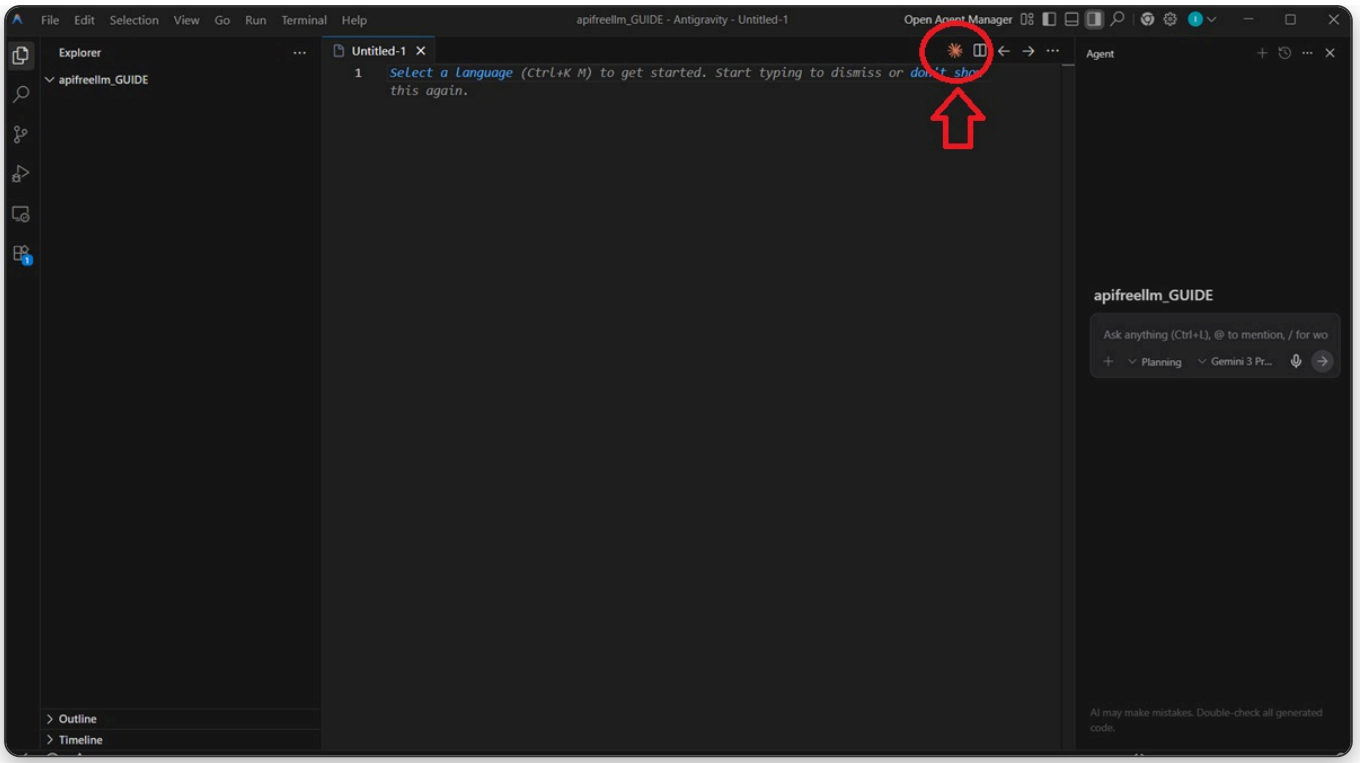
Claude Code als VS Code-Erweiterung installieren

Jetzt, da der Editor offen ist, ist es Zeit, Claude Code zu installieren. Da der Editor auf VS Code basiert, hast du Zugriff auf den Extensions-Marketplace. Klicke auf das **Extensions-Symbol** in der linken Seitenleiste (oder drücke `Ctrl+Shift+X`). Gib in der Suchleiste **„claude code"** ein. Du musst möglicherweise durch die Ergebnisse scrollen — suche nach **„Claude Code for VS Code"** von Anthropic. Sobald du es gefunden hast, klicke auf den **Install**-Button.



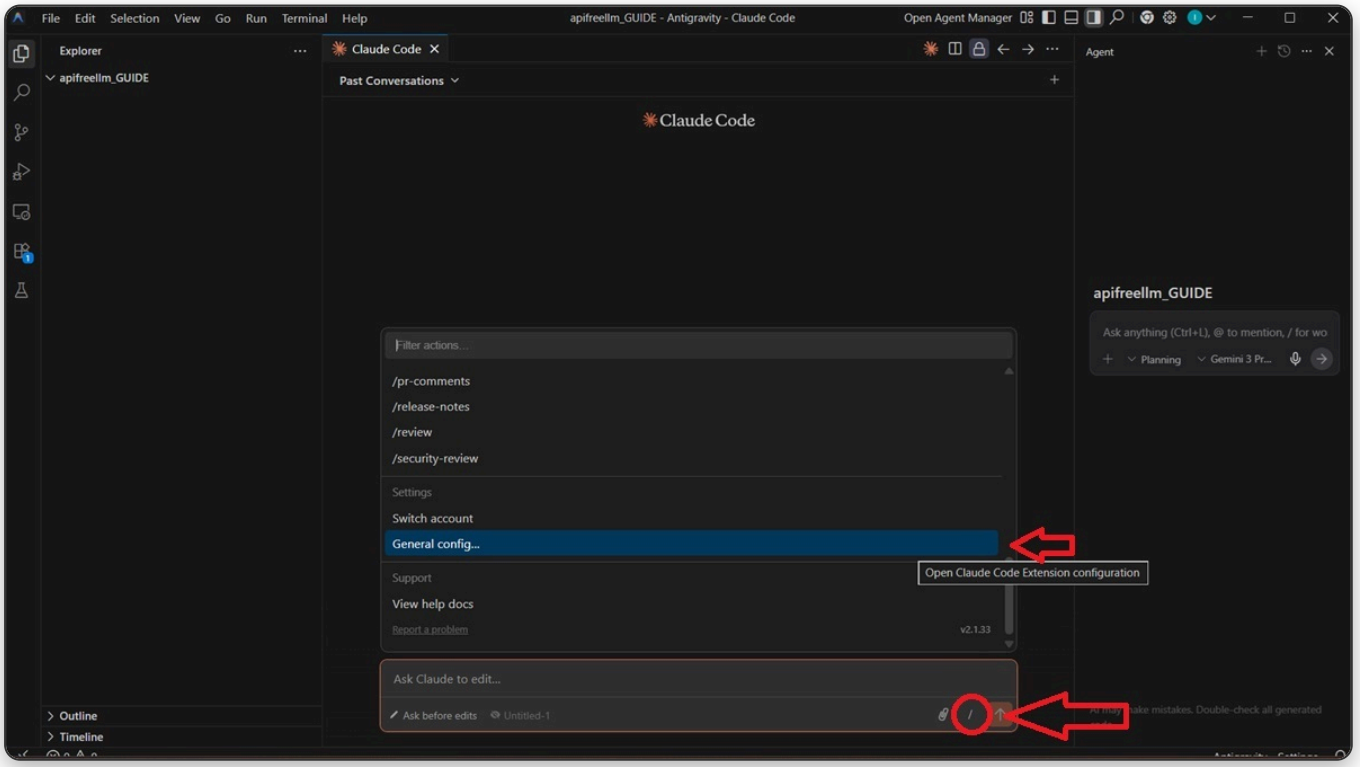
Suche nach „claude code" im Extensions-Marketplace, scrolle nach unten und klicke auf Install.

Nach der Installation schließe das Extensions-Panel und öffne eine neue Datei (oder eine beliebige Datei in deinem Projekt). Du wirst ein kleines **Claude Code-Symbol** bemerken, das oben rechts im Editor erscheint — es sieht aus wie ein kleines oranges Symbol. Klicke darauf, um das Claude Code-Chat-Panel zu öffnen.



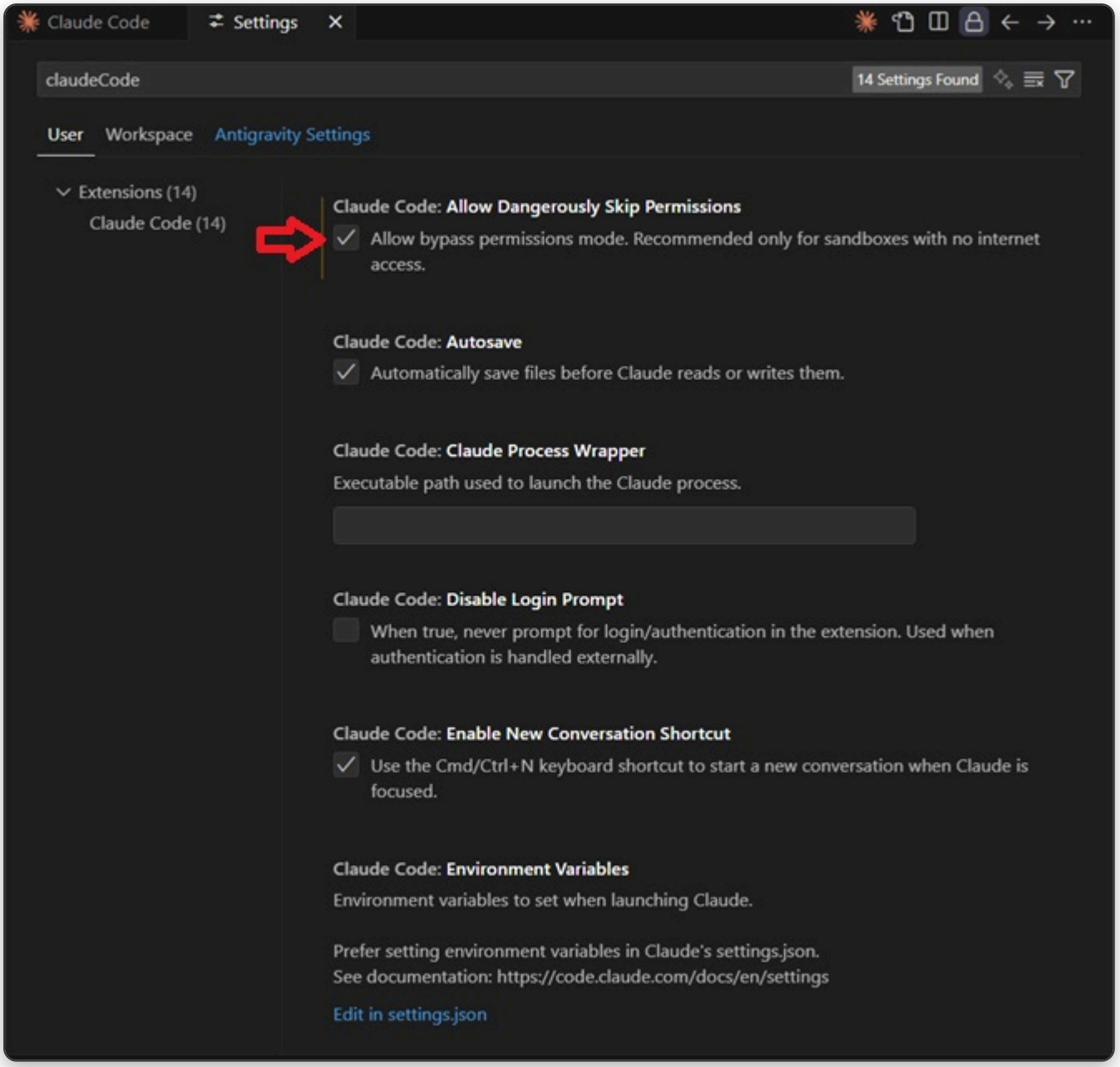
Der Claude Code-Button (eingekreist) erscheint oben rechts im Editor. Klicke darauf, um den Claude Code-Chat zu öffnen.

Claude Code wird dich bitten, dich mit deinem Anthropic-Account anzumelden (dem mit deinem Pro-Abonnement). Nach der Anmeldung musst du es konfigurieren. Im Claude Code-Chat-Panel klicke auf den **/**-Button unten im Panel. Ein Menü erscheint mit verschiedenen Befehlen. Scrolle nach unten zum Abschnitt **Settings** und klicke auf **„General config..."**, um die Claude Code-Erweiterungskonfiguration zu öffnen.



Klicke auf den „/"-Button unten, scrolle dann zu Settings und wähle „General config...", um die Konfiguration zu öffnen.

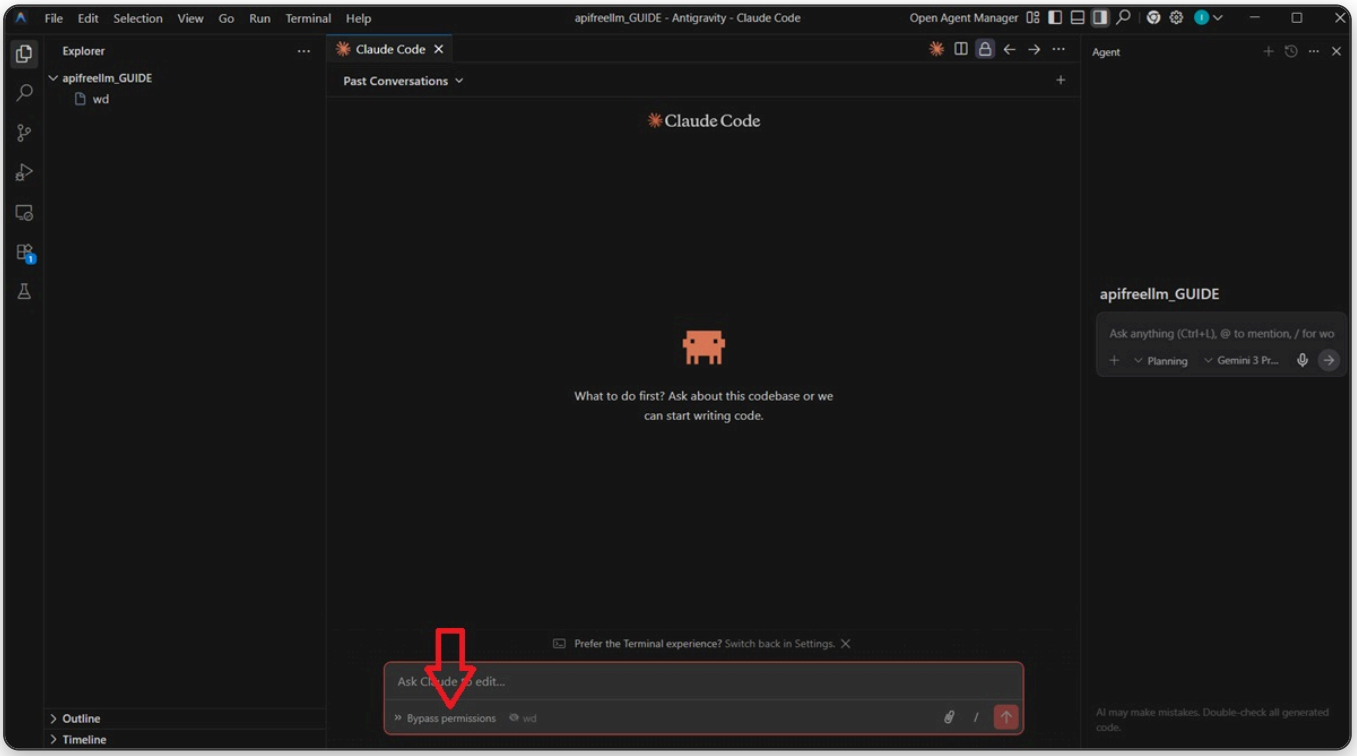
Im sich öffnenden Konfigurationspanel suche nach der Option **„Allow Dangerously Skip Permissions"**. Aktiviere diesen Schalter. Sobald aktiviert, kannst du **„Bypass permissions"** als deinen Berechtigungsmodus auswählen, der es Claude Code erlaubt, vollständig autonom zu arbeiten — Dateien lesen, Code schreiben, Terminalbefehle ausführen und Änderungen vornehmen, ohne bei jedem Schritt nach Bestätigung zu fragen.



Aktiviere den „Allow Dangerously Skip Permissions"-Schalter in den Claude Code-Einstellungen und wähle dann „Bypass permissions".

> **Wichtig:** Mit aktivierten Bypass-Berechtigungen führt Claude Code Aktionen auf deinem System ohne manuelle Genehmigung aus. Das ist essentiell für einen flüssigen Entwicklungs-Workflow, bringt aber Verantwortung mit sich. Sei vorsichtig mit dem Code, den du ausführen lässt, und richte ihn niemals auf nicht vertrauenswürdige Repositories oder verdächtige URLs. KI-Agenten können durch Prompt-Injection ausgenutzt werden — bösartige Inhalte, die in Dateien oder Websites versteckt sind und den Agenten dazu verleiten, schädliche Befehle auszuführen. Überprüfe immer, was Claude Code tut, besonders bei der Arbeit mit externen Ressourcen.

Wir empfehlen außerdem, die Einstellung zu aktivieren, die **„Bypass permissions" als Standardauswahl** für neue Chats festlegt, damit du es nicht jedes Mal manuell auswählen musst, wenn du eine neue Konversation startest. Schließe jetzt Claude Code und öffne es erneut (indem du wieder auf den Claude Code-Icon-Button klickst). Ab jetzt siehst du die Option **„Bypass permissions"** im Berechtigungswähler-Button unten im Claude Code-Chat-Panel. Klicke darauf, um sie zu aktivieren.



Wähle „Bypass permissions" aus dem im Screenshot angezeigten Button.

Mit aktivierten Bypass-Berechtigungen führt Claude Code Befehle aus, erstellt Dateien, bearbeitet Code und führt Terminal-Operationen komplett eigenständig durch — ohne bei jedem Schritt nach deiner Genehmigung zu fragen. Das ist es, was dir ermöglicht, **den gesamten Entwicklungs-Workflow zu 100% zu automatisieren**: Du beschreibst, was du bauen willst, und Claude Code baut es autonom von Anfang bis Ende. Kein ständiges Klicken auf „Accept" bei jeder einzelnen Dateiänderung oder Befehlsausführung mehr. Du gibst die Anweisungen, und der Agent erledigt den Rest.

Dein Kontingent clever verwalten

Etwas, das du von Anfang an wissen solltest: Jede Interaktion mit Claude Code verbraucht **Tokens**, und dein Abonnement hat ein Nutzungslimit. Die gute Nachricht ist, dass du genau überwachen kannst, wie viel du verbraucht hast. Im Claude Code-Chat-Panel klicke auf den **/**-Button — unter den verfügbaren Befehlen findest du deine **aktuelle Nutzung**, die zeigt, wie viele Tokens du verbraucht hast und wie viel Kapazität du noch hast.

Nicht alle Modelle verbrauchen Tokens gleich schnell. **Claude Opus 4.6** ist das intelligenteste und leistungsfähigste Modell, verbraucht aber auch die meisten Tokens pro Interaktion. Kleinere Modelle wie **Sonnet** oder **Haiku** sind weniger leistungsfähig, haben aber höhere Nutzungslimits und verbrauchen deutlich weniger Tokens. Unsere Empfehlung: Verwende **Opus für komplexe Aufgaben**, die tiefes Reasoning, Änderungen über mehrere Dateien oder architektonische Entscheidungen erfordern — hier macht seine Intelligenz einen echten Unterschied. Für einfachere Aufgaben wie schnelle Fixes, kleine Bearbeitungen oder unkomplizierte Fragen wechsle zu einem leichteren Modell, um deine Opus-Tokens für die Momente aufzusparen, in denen sie wirklich zählen.

Es gibt noch eine weitere Strategie, um deine Claude-Tokens zu sparen: **Nutze den eingebauten Agenten von Antigravity** für Fragen, die nicht Claudes Intelligenzniveau erfordern. Antigravity unterstützt mehrere Modelle, aber wir empfehlen für diese schnellen Fragen **Gemini** — da Antigravity ein Google-Produkt ist, hat Gemini das höchste Nutzungslimit und ist auch das schnellste Modell auf der Plattform. Brauchst du eine schnelle CSS-Erinnerung? Willst du die Syntax für einen Git-Befehl wissen? Neugierig, wie eine Bibliothek funktioniert? Frag Antigravity statt Claude Code. So sparst du deine Claude-Tokens für die schwere Entwicklungsarbeit, wo sie den größten Unterschied machen.

Wie im früheren Screenshot zu sehen, empfehlen wir, den **Claude Code-Chat links** und den **Antigravity-Agenten-Chat rechts** zu platzieren. Dieses Seite-an-Seite-Layout gibt dir jederzeit sofortigen Zugriff auf beide Agenten.

> Der smarte Workflow: **Opus zum Bauen, leichtere Modelle für schnelle Aufgaben, Antigravity für allgemeine Fragen.** So maximierst du den Wert deines Claude-Abonnements. Wenn dein Claude-Nutzungslimit knapp wird, denk daran, dass du immer Antigravitys Gemini-Agenten direkt neben dir hast für einfachere Fragen — nutze ihn, um deine Claude-Tokens für die Entwicklungsaufgaben aufzusparen, die sie wirklich brauchen.

Grundlegende Konfiguration

Unabhängig davon, wie du Claude Code installierst, werden diese Konfigurationsschritte deine Ergebnisse dramatisch verbessern:

- **Verwende eine CLAUDE.md-Datei** — Platziere eine `CLAUDE.md`-Datei im Stammverzeichnis deines Projekts. Diese Datei wird von Claude Code automatisch zu Beginn jeder Sitzung gelesen. Verwende sie, um deine Projektstruktur, Coding-Konventionen, den Tech-Stack und alle Regeln zu beschreiben, denen der Agent folgen soll. Stell sie dir als Onboarding-Dokumentation für deinen KI-Entwickler vor.

- **Halte dein Projekt organisiert** — KI-Agenten arbeiten dramatisch besser mit sauberen, gut strukturierten Codebasen. Wenn dein Code chaotisch ist, wird der Agent chaotische Ausgabe produzieren. Gute Ordnerstruktur, klare Namenskonventionen und konsistente Muster machen einen enormen Unterschied.
- **Verwende Versionskontrolle** — Arbeite immer mit initialisiertem Git. Das gibt dir ein Sicherheitsnetz. Wenn der Agent einen Fehler macht, kannst du sofort zurücksetzen. Es ermöglicht dem Agenten auch, Commits für dich zu erstellen, was überraschend nützlich ist, um nachzuverfolgen, was sich geändert hat und warum.

Unverzichtbare Tools: Git & GitHub CLI

Bevor wir anfangen, irgendetwas zu bauen, müssen zwei Tools auf deinem System installiert sein: **Git** und die **GitHub CLI (gh)**. Diese sind fundamental für jeden modernen Entwicklungs-Workflow und ermöglichen es deinen KI-Agenten, deine Code-Repositories autonom zu verwalten.

Warum Git und GitHub CLI wichtig sind

Git ist das Versionskontrollsystem, das jede Änderung in deinem Projekt verfolgt. Es ist dein Sicherheitsnetz: Wenn Claude Code einen Fehler macht, kannst du sofort zurücksetzen. Es ermöglicht dem Agenten auch, Commits zu erstellen, Branches zu verwalten und eine saubere Historie der Projektentwicklung zu führen — alles automatisch.

GitHub CLI (gh) ist ein Kommandozeilen-Tool, das direkten Zugang zu GitHub vom Terminal aus bietet. Das ist der Schlüssel zur vollen Automatisierung: Sobald `gh` installiert und authentifiziert ist, kann Claude Code Repositories erstellen, Code pushen, Pull Requests verwalten, Repository-Einstellungen konfigurieren, GitHub Actions für das Deployment einrichten, Secrets hinzufügen und vieles mehr — alles von seinem Terminal aus, ohne dass du GitHub im Browser öffnen musst.

Git und GitHub CLI installieren

Der einfachste Weg, diese Tools zu installieren, ist, **Claude Code oder Antigravity zu bitten, es für dich zu tun**. Sag deinem Agenten einfach: *„Installiere Git und die GitHub CLI auf meinem System."* Der Agent erkennt dein Betriebssystem und führt die entsprechenden Installationsbefehle aus. Unter Windows wird er typischerweise `winget` verwenden oder die Installer herunterladen; auf macOS wird er `brew` verwenden; auf Linux `apt` oder den Paketmanager deines Systems.

Wenn du sie lieber manuell installieren möchtest, kannst du Git von der offiziellen Website und die GitHub CLI von der GitHub-Releases-Seite herunterladen. Aber es von der KI erledigen zu lassen ist schneller und vermeidet häufige Installationsfehler.

GitHub CLI authentifizieren

Nach der Installation musst du dich einloggen, damit `gh` auf deinen GitHub-Account zugreifen kann. Auch hier kannst du einfach deinen Agenten fragen: *„Logge mich in die GitHub CLI ein."* Der Agent führt `gh auth login` aus und führt dich durch den Authentifizierungsprozess, der typischerweise das Öffnen eines Browser-Links und die Eingabe eines Codes umfasst. Sobald authentifiziert, hat der Agent vollen Zugriff auf deine GitHub-Repositories.

> **Das ist ein Game-Changer.** Mit authentifiziertem `gh` kannst du Claude Code Dinge sagen wie: *„Erstelle ein neues privates Repository namens my-app, initialisiere das Projekt und pushe den Code."* Oder später: *„Richte eine GitHub Action ein, die bei jedem Push auf Main auf meinen Server deployt."* Der Agent erledigt alles — Dateien erstellen, Secrets konfigurieren, Workflows einrichten — ohne dass du den Editor verlassen musst. So sieht echte Entwicklungsautomatisierung aus.

3. Dein erstes Projekt: Von Null auf Live

> *Deine Umgebung ist bereit. Claude Code ist offen, Antigravity läuft, Git und GitHub CLI sind konfiguriert. Es ist Zeit, etwas Echtes zu bauen. Ab hier wechselt der Kurs vom Setup zur Strategie — praktische Techniken und Erkenntnisse, die dir beim Bauen mit KI-Agenten einen echten Vorteil verschaffen.*

Nie bei Null anfangen

Das ist der wichtigste Ratschlag in diesem gesamten Kurs: **Fang niemals bei Null an.**

Auch wenn Claude Opus ein unglaublich fortschrittliches und intelligentes Modell ist, wird es Fehler machen. Jedes KI-Modell tut das. Es könnte deine Projektstruktur falsch verstehen, eine veraltete Bibliothek verwenden, ein inkonsistentes Dateilayout erstellen oder Code generieren, der nicht ganz zusammenpasst, wenn das Projekt wächst. Von einem leeren Ordner zu starten bedeutet, dass die KI Hunderte von Entscheidungen ohne Bezugspunkt treffen muss — und einige davon werden unweigerlich falsch sein.

Hier ist die Realität: **99,9% von dem, was du bauen willst, existiert bereits** in irgendeiner Form. Ob es ein E-Commerce-Shop, ein SaaS-Dashboard, eine Portfolio-Website, eine Social-Media-App oder eine Buchungsplattform ist — jemand hat bereits etwas Ähnliches gebaut. Und viele dieser Projekte sind Open-Source, als Vorlagen auf GitHub verfügbar, bereit zum Klonen und Anpassen.

Dein Workflow sollte immer gleich beginnen: **Suche zuerst nach einer Vorlage.** Geh auf GitHub und suche nach Open-Source-Projekten, die zu dem passen, was du bauen willst. Finde eines, das deiner Vision nahekommt, klone es in deinen Projektordner, und richte dann Claude Code darauf aus. Statt von Null zu bauen, modifiziert, verbessert und individualisiert der Agent jetzt eine bestehende, funktionierende Codebasis. Der Unterschied in Qualität und Geschwindigkeit ist enorm.

> **Vorlagen sind kein Schummeln — sie sind Strategie.** Professionelle Entwickler verwenden ständig Boilerplates und Starter-Kits. Du kopierst nicht jemandes Produkt. Du verwendest ein strukturelles Fundament und baust dein eigenes einzigartiges Produkt darauf auf. Die KI performt dramatisch besser, wenn sie bestehenden Mustern folgen kann, statt alles aus dem Nichts zu erfinden.

Den richtigen Stack wählen

Wenn deine Vorlagensuche kein Ergebnis bringt und du wirklich ein neues Projekt starten musst, kann die Technologie, die du wählst, einen Unterschied machen — besonders bei der Arbeit mit KI-Agenten. Allerdings gibt es keine einzig richtige Wahl. Der beste Stack ist der, der dich am schnellsten zu einem funktionierenden Produkt bringt.

Claude Code ist, wie alle LLMs, grundsätzlich besser darin, **webbasierten Code** zu schreiben: HTML, CSS, JavaScript und TypeScript. Das ist die Sprache des Internets, und darauf wurden diese Modelle am meisten trainiert. Je näher dein Projekt bei Web-Technologien bleibt, desto besser wird die KI performen.

Für **Webanwendungen** ist **Next.js** eine ausgezeichnete Wahl, wenn du es finden kannst. Es ist ein modernes React-Framework mit eingebautem Server-Side Rendering, das entscheidend für SEO ist. Claude Code funktioniert außergewöhnlich gut mit Next.js-Projekten: Es versteht das dateibasierte Routing, API-Routen, Server-Komponenten und das gesamte Ökosystem. Du wirst eine enorme Anzahl von Next.js-Vorlagen auf GitHub finden, für praktisch jede Art von Anwendung.

Für **Desktop-Anwendungen** (ausführbare Dateien für Windows, macOS oder Linux) ist **Electron** eine großartige Option. Electron ermöglicht es dir, Desktop-Apps mit HTML, CSS und JavaScript zu bauen — denselben Web-Technologien, in denen Claude hervorragend ist. Da die Benutzeroberfläche im Grunde eine Webseite ist, die in einem nativen Fenster gerendert wird, kann die KI schöne, funktionale Desktop-Anwendungen mit derselben Leichtigkeit wie eine Website erstellen.

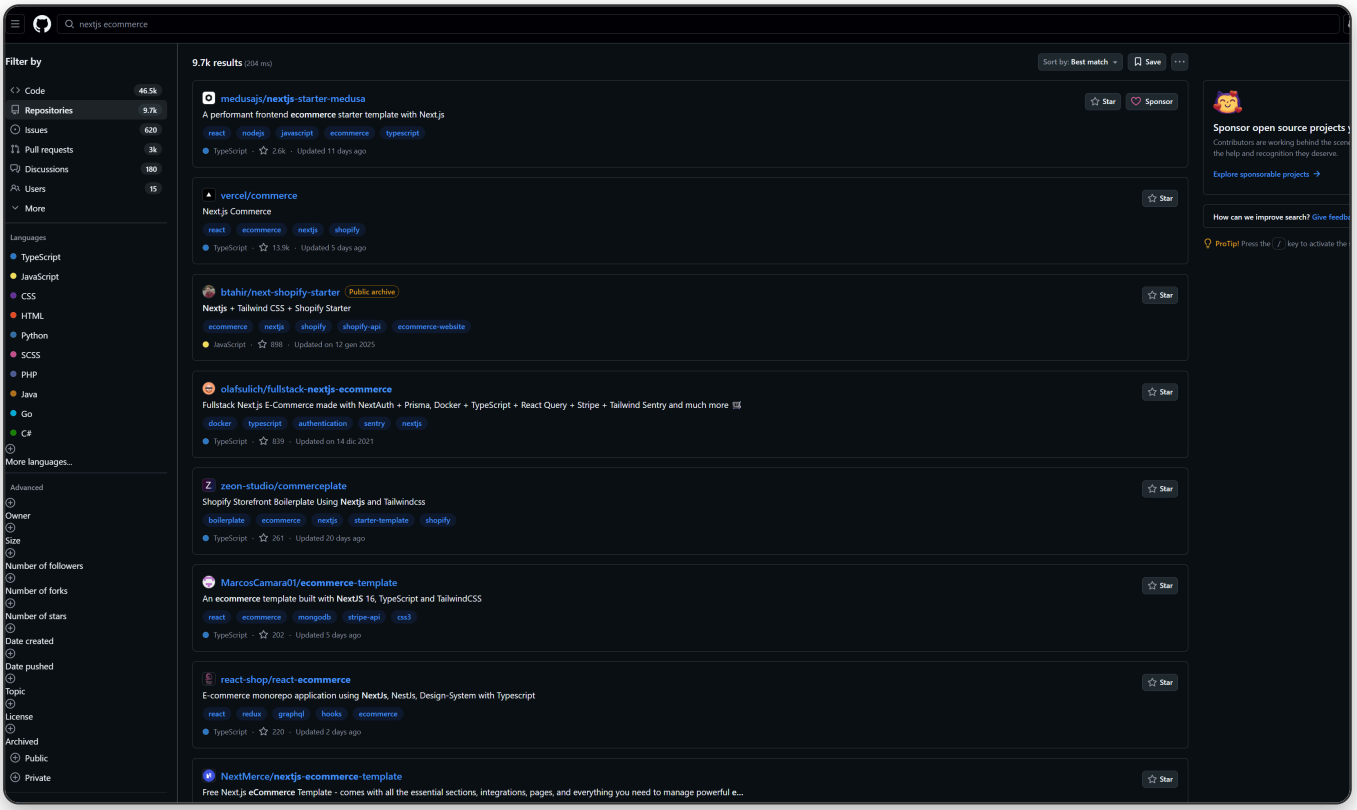
Aber hier ist die wichtige Nuance: **Eine gut gebaute Vorlage in einem älteren Stack ist fast immer besser als bei Null mit einem modernen anzufangen.** Du könntest ein PHP-, jQuery- oder Laravel-Projekt finden, das genau das ist, was du brauchst — komplett, gut strukturiert, praxiserprobt, mit allen Features bereits implementiert. In dem Fall: Verwende es. KI-Agenten können mit jeder Technologie arbeiten, und die Zeitersparnis durch ein solides, fertiges Fundament überwiegt bei Weitem die theoretischen Vorteile eines neueren Frameworks. Claude Code kann PHP-, Python-, Ruby- oder jede andere Codebasis problemlos verstehen und modifizieren.

Die Priorität ist einfach: **Finde die beste verfügbare Vorlage.** Wenn du zwei Vorlagen ähnlicher Qualität findest und eine Next.js verwendet, während die andere PHP nutzt, nimm Next.js. Aber wenn die PHP-Vorlage vollständiger, funktionsreicher und besser gepflegt ist — wähle diese ohne zu zögern. Qualität und Vollständigkeit des Ausgangspunkts zählen mehr als die Modernität des Stacks.

> **Die Faustregel:** Verwende, was immer dich deinem Ziel am nächsten bringt. Strebe moderne Web-Technologien an (Next.js, Electron, React Native), wenn verfügbar, aber lehne niemals eine großartige Vorlage ab, nur weil sie einen älteren Stack verwendet. Die Zeitersparnis durch ein solides Fundament ist immer wertvoller als Stack-Reinheit.

Praktisches Beispiel: Eine Vorlage finden

Nehmen wir an, du willst einen E-Commerce-Shop bauen. Statt Claude Code zu sagen *„Baue mir eine E-Commerce-Seite von Grund auf"*, öffne GitHub und suche nach etwas wie **„nextjs ecommerce"**. Du wirst Dutzende fertige Projekte mit Produktlisten, Warenkörben, Checkout-Abläufen und Zahlungsintegration finden, die bereits gebaut sind.



Eine schnelle GitHub-Suche nach „nextjs ecommerce" zeigt bereits mehrere vielversprechende Vorlagen.

Etwas Wichtiges, das du im Hinterkopf behalten solltest: Viele Vorlagen auf GitHub sind **Freemium-Projekte** — der Kern ist Open-Source und kostenlos, aber einige Features oder Premium-Versionen erfordern eine Bezahlung. Überprüfe immer die README und die Lizenz des Repositorys, bevor du dich für eine Vorlage entscheidest. Vermeide alles, das bezahlte Abonnements oder versteckte Kosten erfordert, die du nicht brauchst.

Wenn wir die Suchergebnisse betrachten, fällt **fullstack-nextjs-ecommerce** auf — und es ist ein ausgezeichneter Ausgangspunkt. Analysieren wir warum. Das Projekt verwendet Next.js mit TypeScript, was genau das ist, was wir für KI-gestützte Entwicklung wollen. Aber was wirklich herausstich, ist, was bereits integriert ist: **Stripe für Zahlungen**, **PostgreSQL mit Prisma** als Datenbank und **NextAuth für Authentifizierung**. Das sind drei der kritischsten — und fehleranfälligsten — Teile jeder Webanwendung.

Dass **Stripe bereits integriert** ist, ist besonders wertvoll. Zahlungsabwicklung umfasst Webhooks, Session-Management, Fehlerbehandlung und Randfälle, die überraschend knifflig sein können. Wenn etwas bei Zahlungen schiefgeht, sind deine Kunden die Leidtragenden — fehlgeschlagene Abbuchungen, doppelte Zahlungen oder defekte Checkout-Abläufe sind die Art von Bugs, die Vertrauen zerstören und dich echtes Geld kosten. Mit einer Vorlage zu starten, die bereits eine funktionierende Stripe-Integration hat, erspart dir diese Kopfschmerzen.

Das Projekt verwendet auch **PostgreSQL** als Datenbank, was eine solide Wahl ist. Postgres ist zuverlässig, gut dokumentiert, bietet etwas bessere Sicherheitsstandards im Vergleich zu Alternativen wie MySQL und kann leicht auf Amazon AWS, Hetzner oder ähnlichen Anbietern gehostet werden — Server-Setup und Deployment behandeln wir im nächsten Kapitel. Dass **NextAuth** bereits verdrahtet ist, bedeutet, dass die Benutzer-Authentifizierung direkt einsatzbereit ist, ein weiteres komplexes Puzzleteil, das du nicht von Grund auf bauen musst.

Das ist die Kraft, mit der richtigen Vorlage zu starten: Statt Tage (oder Wochen) damit zu verbringen, Zahlungen, Datenbank und Authentifizierung einzurichten — alles Dinge, bei denen leicht etwas schiefgehen kann — startest du mit einem Projekt, in dem diese kritischen Systeme bereits funktionieren. Du kannst dich dann voll darauf konzentrieren, das Produkt nach deiner Vision anzupassen: das Design ändern, deine Produkte hinzufügen, die Geschäftslogik modifizieren und die Features bauen, die deinen Shop einzigartig machen.

Sobald du deine Vorlage gefunden hast, ist der nächste Schritt einfach: **Lade sie herunter und platziere sie in deinem Workspace-Ordner.** Öffne diesen Ordner mit Claude Code (oder Antigravity) und fang an zu arbeiten. Du kannst Claude Code anweisen, die Vorlage direkt zu modifizieren — das Branding ändern, neue Seiten hinzufügen, das Layout überarbeiten, neue Features integrieren — oder du kannst die Vorlage als **Referenz** für dein eigenes Projekt verwenden. Selbst wenn du etwas leicht Anderes baust, bedeutet die Vorlage im selben Workspace, dass Claude Code sie ansehen, die Code-Muster studieren und als Grundlage für das verwenden kann, was er schreibt.

Das ist ein entscheidender Punkt: **Gib Claude Code immer etwas als Referenz.** Wenn der Agent eine solide, funktionierende Codebasis als Vorlage hat, schreibt er deutlich besseren Code. Er folgt denselben Mustern, verwendet dieselben Konventionen und produziert konsistente und zuverlässige Ausgabe. Wenn er nichts als Referenz hat und alles von Grund auf generieren muss, passieren Fehler — inkonsistente Dateistrukturen, falsche Bibliotheksversionen, Code der nicht zusammenpasst. Eine Vorlage wirkt als Anker, der die KI auf Kurs hält und hochwertige, kohärente Ergebnisse liefert.

Was, wenn die Vorlage keine Zahlungen oder Datenbank hat? Das ist auch in Ordnung. Dieser Kurs enthält fertige Stripe-Integrationsdateien, die du Claude Code direkt geben kannst, um Zahlungen in jedes Projekt zu integrieren. Für die Datenbank empfehlen wir PostgreSQL oder welche Datenbank die Vorlage bereits verwendet — kämpfe nicht gegen die Entscheidungen der Vorlage an, es sei denn, es gibt einen triftigen Grund. Das Gleiche gilt für die Authentifizierung: Wenn die Vorlage NextAuth, Clerk oder ein anderes Auth-System verwendet, arbeite damit. Das Ziel ist, das bereits Gebaute zu nutzen, nicht alles zu ersetzen.

4. Vom Code zur Produktion

Dein Produkt ist gebaut. Jetzt muss es Zahlungen akzeptieren, auf einem Server laufen und schnell und sicher für Nutzer weltweit sein. Dieses Kapitel behandelt die wesentlichen Services, die dein Projekt von lokalem Code in ein live geschaltetes, produktionsreifes Business verwandeln.

Ein Hinweis zu diesem Kapitel. Wir halten die Dinge hier knapp und auf den Punkt. Unser Ziel ist es, dir zu sagen, **was** du tun musst und **warum** — nicht lange Tutorials zu schreiben, die deine Zeit verschwenden. Jedes LLM (Claude, Gemini, ChatGPT) kann die Details erklären, dich durch jeden Schritt führen und deine Fragen weit besser beantworten als eine statische Seite. Wann immer etwas unklar ist, frag einfach deinen Agenten: *„Ich folge einem Kurs und dort steht, dass ich [X machen] muss. Kannst du mir erklären, was das bedeutet und mich durchführen?"* Das ist schneller, persönlicher und immer aktuell.

Zahlungen mit Stripe

Wenn dein Produkt etwas verkauft, brauchst du einen Zahlungsanbieter. **Stripe** ist der Industriestandard: zuverlässig, gut dokumentiert und von praktisch jedem KI-Agenten unterstützt, weil es so weit verbreitet ist.

Folgendes musst du tun:

- **Erstelle ein Stripe-Konto** auf stripe.com. Du bekommst **API-Schlüssel** (einen öffentlichen Schlüssel für das Frontend, einen geheimen Schlüssel für das Backend) und Zugang sowohl zum **Testmodus** (simulierte Zahlungen für die Entwicklung) als auch zum **Live-Modus** (echtes Geld).
- **Richte Webhooks** im Stripe-Dashboard ein. Webhooks sind URLs auf deinem Server, die Stripe aufruft, wenn etwas passiert — eine Zahlung ist erfolgreich, ein Abonnement wird verlängert, eine Abbuchung schlägt fehl. So erfährt deine App, wann ein Abonnement aktiviert,

eine Bestellung bestätigt oder ein Fehler behandelt werden muss. Du brauchst Webhooks sowohl für **lokales Testen** (Stripe hat ein CLI-Tool, das Events an deinen Localhost weiterleitet) als auch für die **Produktion** (die auf deinen Live-Server zeigen). Bring immer erst alles lokal zum Laufen, bevor du live gehst.

- **Integriere Stripe in dein Projekt.** Wenn deine Vorlage bereits Stripe hat, trage einfach deine API-Schlüssel ein. Wenn nicht, enthält dieser Kurs fertige Stripe-Integrationsdateien — lege sie in deinen Workspace und weise Claude Code an, sie zu integrieren. Stripe unterstützt einmalige Zahlungen und wiederkehrende Abonnements, beides mit gehosteten Checkout-Seiten, die Kartenvalidierung, 3D Secure und PCI-Compliance für dich übernehmen.

Eine entscheidende Regel: **Verifiziere Zahlungen immer serverseitig über Webhooks**, vertraue niemals dem Frontend. Der Webhook ist Stripe, der deinem Server direkt mitteilt, dass tatsächlich Geld geflossen ist — es ist die einzige zuverlässige Wahrheitsquelle.

Hosting: AWS, Hetzner & mehr

Deine App muss irgendwo leben. Die gute Nachricht: **AWS gibt dir ein Free Tier**, wenn du dich anmeldest — für **ein volles Jahr** bekommst du einen **t2.micro-Server** und eine **Micro-Datenbank** komplett kostenlos. Das reicht, um dein erstes Projekt zu hosten, während du die Idee validierst und erste Nutzer gewinnst.

Du weiβt nicht, wie du ein AWS-Konto einrichtest oder das Free Tier nutzt? Frag einfach Claude oder Antigravity — sie führen dich durch jeden Schritt.

Wenn du über das Free Tier hinauswächst, hier ist, was du über Server-Dimensionierung wissen musst:

- **t3.small** ist der Sweet Spot für die meisten Apps — gute CPU, genug RAM und vernünftige Preise (~16$/Monat auf AWS). Das „t" bezieht sich auf die Instanzgeneration; ältere Generationen (t2 usw.) können manchmal mehr kosten für weniger Leistung, also bleib bei der neuesten verfügbaren.
- **Hetzner** ist eine europäische Alternative, die *deutlich* günstiger ist — du bekommst Leistung vergleichbar mit einem t3.small für ungefähr **4$/Monat**. Das ist ungefähr 4× günstiger als AWS für ähnliche Spezifikationen.
- **Nicht jede App braucht einen Server.** Wenn dein Projekt eine statische Seite oder eine JAMstack-App ist, brauchst du möglicherweise gar keinen dedizierten Server. Plattformen wie Vercel, Netlify oder sogar Cloudflare Pages können es kostenlos oder nahezu kostenlos hosten. Frag immer dein LLM: *„Was ist das beste Hosting für meine spezifische App?"*

Unsere Empfehlung: **Starte mit dem AWS Free Tier** für dein erstes Jahr und evaluiere dann, ob Hetzner oder ein anderer Anbieter für dein Budget und den Standort deiner Nutzer sinnvoller ist. Wenn die meisten deiner Nutzer in Europa sind, geben dir Hetzners deutsche Server eine geringere Latenz. Wenn dein Publikum global oder US-basiert ist, könnten AWS-Regionen besser passen.

SSH-Schlüssel & KI-Serververwaltung. Damit Claude Code sich mit deinem Server verbinden und ihn remote verwalten kann, musst du einen **SSH-Schlüssel** einrichten. Bitte Claude, einen zu generieren und auf deinem Server zu konfigurieren. Einmal verbunden, kann Claude Code deployen, Services verwalten, Probleme beheben — alles von deinem Terminal. Für Serververwaltungsaufgaben empfehlen wir **Opus** für die besten Ergebnisse, obwohl Sonnet auch funktioniert, wenn du Tokens sparen willst (mit einer etwas höheren Fehlerwahrscheinlichkeit).

Cloudflare: Performance & Schutz

Sobald deine App auf einem Server läuft, brauchst du etwas, das davor sitzt, um sie zu schützen und zu beschleunigen. Das ist **Cloudflare**. Die gute Nachricht: **Der kostenlose Plan reicht für die überwiegende Mehrheit der Websites völlig aus.** Du brauchst keinen bezahlten Plan.

Aber zuerst brauchst du einen **Domainnamen**. Wir empfehlen, einen direkt bei **Cloudflare** oder **Namecheap** zu kaufen — beide sind zuverlässig und fair bepreist. Sobald du deine Domain hast, musst du das **DNS** konfigurieren, damit es auf die IP-Adresse deines AWS- oder Hetzner-Servers zeigt. Frag einfach dein LLM: *„Ich habe eine Domain bei [Cloudflare/Namecheap] gekauft. Wie richte ich DNS ein, damit es auf meinen Server unter [deine IP] zeigt?"* Es wird dich durchführen.

Warum ist Cloudflare so wichtig? Es schützt deine Seite vor **DDoS-Angriffen**, verschiedenen **Sicherheitslücken** und gängigen Web-Schwachstellen. Es bietet auch einen **globalen Cache**, was bedeutet, dass deine Inhalte von Servern in der Nähe deiner Nutzer ausgeliefert werden, was deine Seite weltweit schneller macht. Ohne einen Service wie Cloudflare setzt du deine Seite ernsthaften Risiken aus — besonders jetzt, im KI-Zeitalter, wo jeder KI-Tools verwenden kann, um Schwachstellen zu finden, Fehlkonfigurationen auszunutzen oder sogar Daten von schlecht geschützten Websites zu stehlen. Überspring das nicht.

Schneller Tipp: Flexible SSL-Modus. Bei der Einrichtung von Cloudflare kannst du den **Flexible** SSL-Modus wählen. Das bedeutet, die Verbindung zwischen Cloudflare und deinem Server verwendet einfaches HTTP, aber die Verbindung zwischen Cloudflare und deinen Endnutzern ist HTTPS — Besucher sehen also ein sicheres Schloss-Symbol. Das erspart dir die Konfiguration von SSL-Zertifikaten auf deinem Server, was großartig für den schnellen Einstieg ist. Für Produktions-Apps, die sensible Daten verarbeiten, solltest du irgendwann auf den **Full**-Modus umsteigen (durchgehend verschlüsselt). Aber für deine ersten Tests und Launches ist Flexible völlig in Ordnung und spart viel Setup-Zeit. Frag dein LLM, um die Unterschiede zu erklären, wenn du unsicher bist, welcher Modus zu deinem Projekt passt.

Cloudflare bietet weit mehr als nur Schutz. Der kostenlose Plan enthält leistungsstarke Funktionen, die viele Entwickler gar nicht kennen:

- **Cloudflare Pages** — kostenloses statisches Hosting für Frontend-Apps (React, Next.js Static Export, Vue, etc.). Du pushst auf GitHub, Cloudflare baut und deployt automatisch. Null

Konfiguration, null Kosten. Perfekt für Landing Pages, Portfolios und JAMstack-Apps.

- **Edge Functions (Workers)** — führe serverlosen Code am Edge aus, nahe bei deinen Nutzern, im kostenlosen Plan. Ideal für API-Routen, Weiterleitungen, A/B-Testing und leichte Backend-Logik ohne dedizierten Server.
- **CDN & Caching** — deine statischen Assets (Bilder, CSS, JS) werden global gecacht, sodass deine Seite von überall auf der Welt blitzschnell lädt.

Die entscheidende Frage ist: **Braucht deine App wirklich einen dedizierten Server, oder kann Cloudflare sie kostenlos hosten?** Frag Claude: *"Ich baue [beschreibe deine App]. Brauche ich einen dedizierten Server auf AWS/Hetzner, oder kann ich Cloudflare Pages und Workers kostenlos nutzen?"* Claude analysiert deinen spezifischen Fall und sagt dir die beste Option. Du könntest überrascht sein, wie viele Projekte vollständig auf Cloudflares kostenlosem Plan laufen können.

API-Keys: Alles automatisieren

Hier ist ein bahnbrechender Tipp, den die meisten Tutorials auslassen: **Erstelle API-Keys für deine Hosting-Provider** und gib sie Claude. So kann Claude deine Infrastruktur direkt vom Terminal aus verwalten — kein Klicken auf Dashboards, kein Kopieren und Einfügen, keine manuelle Arbeit.

- **Cloudflare API-Key** — gehe zu deinem Cloudflare-Dashboard → My Profile → API Tokens → erstelle einen Token. Damit kann Claude automatisch DNS-Einträge konfigurieren, Pages-Projekte einrichten, Workers verwalten, SSL-Einstellungen aktualisieren und vieles mehr. Sag Claude: *"Hier ist mein Cloudflare API-Token. Richte DNS für meine Domain ein, die auf die IP meines Servers zeigt."* In Sekunden erledigt.
- **AWS Access Keys** — gehe zu AWS IAM → erstelle einen Access Key. Damit kann Claude EC2-Instanzen verwalten, Security Groups konfigurieren, RDS-Datenbanken einrichten, S3-Buckets verwalten und deine gesamte AWS-Infrastruktur programmatisch handhaben. Sag Claude: *"Hier sind meine AWS-Zugangsdaten. Starte eine t3.small EC2-Instanz in us-east-1 und konfiguriere die Security Groups für eine Web-App."*
- **Hetzner API-Token** — gehe zu deiner Hetzner Cloud Console → Projekt → Sicherheit → API-Tokens → generiere einen. Claude kann dann Server erstellen, Firewalls konfigurieren, Snapshots verwalten und deine gesamte Hetzner-Infrastruktur handhaben. Sag Claude: *"Hier ist mein Hetzner API-Token. Erstelle einen CX22-Server in Nürnberg mit Ubuntu."*

> **Sicherheitshinweis zu API-Keys.** Diese API-Keys sind mächtig — behandle sie wie Passwörter. **Committe sie niemals in Git**, teile sie nicht öffentlich und füge sie nicht in Chat-Interfaces ein, denen du nicht vertraust. Speichere sie in einer `.env`-Datei oder übergib sie direkt an Claude in deiner Terminal-Sitzung. Wenn ein Key kompromittiert wird, widerrufe ihn sofort im Dashboard des Providers und generiere einen neuen. Verwende beim Erstellen von API-Tokens **immer das Prinzip der geringsten Rechte**: vergib nur die Berechtigungen, die tatsächlich benötigt werden, nicht vollen Admin-Zugang.

Die Kombination dieser API-Keys mit Claude ist unglaublich mächtig. Du kannst buchstäblich sagen: *"Ich habe eine Next.js-App. Deploye sie auf Cloudflare Pages, richte die Custom Domain ein und konfiguriere das DNS — hier sind meine API-Keys."* Und Claude macht alles automatisch. Oder: *"Erstelle eine EC2-Instanz auf AWS, installiere Node.js und PM2, konfiguriere nginx, richte Cloudflare DNS ein und deploye meine App."* Komplettes Infrastruktur-Setup in Minuten, nicht Stunden.

CI/CD mit GitHub Actions

Erinnerst du dich, als wir Git und GitHub CLI in Kapitel 2 eingerichtet haben? Hier zahlt sich alles aus. Mit authentifiziertem `gh` kannst du Claude Code anweisen, dein Projekt in ein **privates GitHub-Repository zu pushen**, **GitHub Actions** einzurichten, alle notwendigen **Secrets** zu konfigurieren (den SSH-Schlüssel deines Servers, Umgebungsvariablen usw.) und eine automatische Deployment-Pipeline zu erstellen — alles von seinem Terminal, ohne dass du die GitHub-Oberfläche anfasst.

Die Idee ist einfach: Sobald GitHub Actions konfiguriert ist, **deployt es jedes Mal automatisch auf deinen Server, wenn Claude Code Code in dein Repository pusht.** Kein manuelles SSH, kein Kopieren von Dateien, kein Ausführen von Befehlen auf dem Server selbst. Claude pusht, GitHub Actions greift es auf, verbindet sich per SSH mit deinem AWS- oder Hetzner-Server und deployt alles. Voll automatisiert.

Sag Claude einfach: *„Pushe dieses Projekt in ein neues privates Repository auf GitHub, richte eine GitHub Action ein, die bei jedem Push auf Main auf meinen Server deployt. Hier sind meine Server-IP und mein SSH-Schlüssel."* Claude weiß bereits, wie das geht — er erstellt die Workflow-Datei, fügt den SSH-Schlüssel und die Server-IP als GitHub-Secrets hinzu und konfiguriert die gesamte Pipeline. Genau dafür haben wir die GitHub CLI vorher installiert.

> **Stolperfalle dynamische IP.** Statische IPs kosten bei AWS und Hetzner üblicherweise extra, also wirst du wahrscheinlich eine **dynamische IP** verwenden, um Geld zu sparen. Das bedeutet, jedes Mal wenn du deinen Server stoppst und neu startest, ändert sich die IP. Wenn das passiert, musst du sie an **zwei Stellen** aktualisieren: im Cloudflare-DNS-Eintrag und im `SERVER_IP`-GitHub-Secret. Weise Claude an, die Server-IP als `SERVER_IP`-Secret in GitHub Actions zu speichern, damit du bei einer Änderung nur eine Variable aktualisieren musst. Sag Claude auch, dass er dich daran erinnern soll, die IP zu überprüfen und zu aktualisieren, wenn ein Deployment fehlschlägt — eine geänderte IP ist fast immer der Grund.

5. Marketing- & SEO-Taktiken

> *Dein Produkt ist live. Jetzt muss die Welt erfahren, dass es existiert. Das effektivste Marketing für Indie-Produkte ist organisch — kostenlos, kreativ und überraschend wirkungsvoll, wenn man es richtig macht. Dieses Kapitel behandelt die exakten Taktiken, die wir verwenden.*

Strategien für organisches Wachstum

Seien wir ehrlich: **Das beste Marketing sieht nicht nach Marketing aus.** Das Internet ist mit Werbung übersättigt, und die Menschen haben einen Reflex entwickelt, alles zu ignorieren, was sich wie eine Promotion anfühlt. Was tatsächlich funktioniert, ist **Stealth-Marketing** — Inhalte, die wie echte Informationen, eine Frage oder eine Empfehlung aussehen statt wie eine Werbeanzeige. Menschen sind von Natur aus neugierig und helfen gerne mit Vorschlägen. Nutze das.

Reddit ist eine der mächtigsten Plattformen dafür. Es ist riesig, wird von Google stark indexiert und ist voller Nischen-Communities, in denen sich deine Zielgruppe bereits aufhält. Aber hier ist der Schlüssel: **Poste niemals aggressive Werbung.** Schreib nicht *„Schaut euch meine tolle neue Seite an!"* — das wird heruntergestimmt, entfernt oder gebannt. Schreib stattdessen etwas wie:

- *„Kann jemand Seiten empfehlen, die ähnlich sind wie [bekannter Konkurrent] oder [deine Seite]? Suche nach Alternativen."*
- *„Hat jemand [deine Produktkategorie] ausprobiert? Ich habe [deine Seite] gefunden, bin aber neugierig auf andere Optionen."*
- Beantworte Fragen in relevanten Subreddits und erwähne dein Produkt dort natürlich, wo es wirklich hilft.

So funktioniert die Mehrheit des erfolgreichen Indie-Marketings auf Reddit. Du lügst nicht — du rahmst dein Produkt als Entdeckung in einem echten Gespräch ein. Menschen klicken, weil sie neugierig sind, nicht weil sie sich verkauft fühlen. Und hier ein Bonus: **Reddit-Posts werden von Google indexiert**, daher kann ein gut platzierter Thread dir Monate oder sogar Jahre lang organischen Suchtraffic bringen.

Du kannst auch einen **Reddit-Post sponsern** mit einem kleinen Budget, um ihn vorübergehend zu boosten. Das pusht ihn höher in den Suchergebnissen, lässt Google ihn schneller indexieren und gibt ihm erste Sichtbarkeit. Frag dein LLM, wie Reddit-Post-Promotion funktioniert und welches Budget sinnvoll ist.

> **Der Funnel-Trick.** Baue auf deiner Startseite oder Landing Page etwas ein, das **Menschen unabhängig von deinem Hauptprodukt anzieht** — ein kostenloses Tool, ein Trendthema, nützliche Informationen oder etwas, das gerade beliebt ist. Das wirkt als **Funnel**: Leute kommen wegen des kostenlosen/interessanten Inhalts, entdecken dein Produkt, und ein Prozentsatz davon konvertiert. Stell es dir als Köder vor, der echten Mehrwert bietet und gleichzeitig zu dem führt, was du verkaufst.

SEO, Content & Distribution

Bevor du mit irgendeinem Marketing anfängst, richte dein **Analytics und Tracking** ein. Du brauchst zwei Dinge:

- **Google Analytics** — Füge den Tracking-Code zu deiner Seite hinzu (bitte Claude, ihn zu integrieren). Es gibt auch eine **Mobile App**, mit der du deinen Traffic jederzeit vom Handy aus überprüfen kannst. Das zeigt dir Gesamtbesuche, Nutzerverhalten, Traffic-Quellen und Conversion-Daten.
- **Google Search Console** — Richte das ein und verbinde es mit Google Analytics. Search Console trackt speziell deinen **organischen Google-Traffic**: Welche Suchanfragen Menschen auf deine Seite bringen, wie oft du in den Suchergebnissen erscheinst und deine Klickraten. Frag dein LLM, wie du beide Tools einrichtest und verbindest.

Wenn du Budget zur Verfügung hast, kann **Google Ads** dir einen Anfangsschub geben. Kurzzeitig Anzeigen zu schalten hilft, Vertrauen bei Googles Algorithmus aufzubauen und bringt frühen Traffic, während dein organisches SEO wächst. Aber die Kosten summieren sich schnell, also betrachte es als kurzfristigen Beschleuniger, nicht als Langzeitstrategie. Dein LLM kann Google Ads-Setup und Budgetierung im Detail erklären.

Für SEO selbst fang mit den Basics an. Öffne die **DevTools** deines Browsers (F12), gehe zu **Lighthouse** und erstelle einen Bericht. Das gibt dir Bewertungen für Performance, Barrierefreiheit, Best Practices und **SEO**. Behebe, was es dir sagt — und ja, du kannst Claude Code bitten, die Fixes zu übernehmen.

Wichtige SEO-Maßnahmen:

- **Füge Übersetzungen hinzu** zu deinen Seiten. Mehrsprachige Unterstützung erhöht deine Reichweite dramatisch. Bitte Claude Code, die Internationalisierung für dein Projekt einzurichten.
- **Füge ordentliche Meta-Tags hinzu** — Titel, Beschreibung, Open Graph-Tags für Social-Media-Sharing, strukturierte Daten. Diese beeinflussen direkt, wie deine Seite in den Google-Suchergebnissen erscheint.
- **Erstelle und übermittle eine Sitemap.** Generiere eine aktuelle Sitemap und lade sie in die Google Search Console hoch. Das sagt Google genau, welche Seiten auf deiner Website existieren, und hilft dabei, dass sie schneller indexiert werden.

SEO braucht Geduld. Erwarte keinen organischen Traffic über Nacht — es dauert Wochen oder Monate, bis Google deine Seiten richtig indexiert und rankt. Aber wenn es einmal greift, ist es **kostenloser Traffic, der immer weiter kommt**, ohne dass du einen Cent ausgibst. Die Klicks, für die du sonst Hunderte Euro über Google Ads bezahlen würdest, kommen kostenlos durch die organische Suche. In der Zwischenzeit investiere die Arbeit auf sozialen Plattformen wie Reddit, um erste Sichtbarkeit und Backlinks aufzubauen. SEO ist ein Langzeitspiel, aber es ist die wertvollste Marketing-Investition, die du machen kannst.

Der aufkommende Trend: KI-gesteuerter Traffic

Es gibt eine neue und schnell wachsende Traffic-Quelle, der die meisten Menschen noch keine Beachtung schenken: **LLMs, die deine Seite empfehlen.** ChatGPT, Gemini, Claude und andere KI-Assistenten werden zunehmend als Suchmaschinen genutzt. Wenn jemand fragt *„Was ist eine gute Seite für [deine Produktkategorie]?"*, können und werden diese Modelle bestimmte Websites empfehlen — und das bringt echten Traffic. Ein erheblicher Teil unserer eigenen Besuche kommt von ChatGPT und anderen LLMs.

Um davon zu profitieren, geh zu deinem **Cloudflare-Dashboard** und stelle sicher, dass du **die Option deaktivierst, die KI-Crawler blockiert**. Viele Seiten blockieren KI-Bots standardmäßig, was verhindert, dass LLMs von deinen Inhalten erfahren. Indem du KI-Crawlern den Zugriff auf deine Seite erlaubst, lässt du diese Modelle deine Seiten indexieren, verstehen was du anbietest, und dich möglicherweise zukünftig Nutzern empfehlen. Das ist im Grunde **kostenloses SEO für das KI-Zeitalter** — und der Boost kann enorm sein.

Denke über Google hinaus. Traditionelles SEO zielt auf die Google-Suche ab. Aber KI-gesteuerte Empfehlungen werden zu einer bedeutenden Traffic-Quelle — und dieser Trend beschleunigt sich nur. Stelle sicher, dass deine Seite für KI-Crawler zugänglich ist, klare und beschreibende Inhalte hat und gut strukturiert ist, damit LLMs leicht verstehen können, was du anbietest. Die Seiten, die sich jetzt für KI-Entdeckung positionieren, werden einen massiven Vorteil haben, wenn dieser Kanal wächst.

Vielen Dank!

Vielen Dank, dass du diesen Kurs gekauft und uns dein Vertrauen, deine Zeit und dein Geld geschenkt hast. Wir hoffen aufrichtig, dass du ihn wertvoll gefunden hast und er dir hilft, etwas Echtes zu bauen.

Wir würden gerne von dir hören. **Tritt unserer Discord-Community bei** auf apifreellm.com — dort sind wir unterwegs, teilen Updates und helfen uns gegenseitig.

Wenn du einen Moment hast, **schreib uns eine Bewertung** und sag uns, was du denkst. Was fandest du am nützlichsten? Was hat gefehlt? Was könnte besser sein? Wir sind wirklich an deinem Feedback interessiert — dieser Kurs ist ein lebendiges Projekt und wir möchten ihn ständig verbessern, basierend auf dem, was **du** tatsächlich brauchst.

Wir sehen uns auf Discord. Jetzt geh und bau etwas Großartiges.